PRIFYSGOL ABERYSTWYTH UNIVERSITY

CS10720 Problems and Solutions

Thomas Jansen

Today: Revision

April 28th

Plans for Today

Exam Revision

Remember CS107 assessment comprises of three elements

1. portfolio 40%
2. in-class test 30%
3. exam 30%

You have **either** already passed
by accumulating $\geq 40\%$ in the portfolio and in-class test
and want to improve your mark
or have not yet passed
and need the exam to pass (and perhaps a bit more)

In both cases you're interested in doing well in the exam

(If you're not interested in doing well in the exam
you're really just wasting your time being here.)

Matrices Using Dynamic Arrays in C

Problem Assume you want to allocate memory for a two-dimensional 10×4 matrix of doubles as one block of memory using malloc.

1. How much memory will be allocated and how do you do that using malloc?
2. How do you access matrix element in the second column in the sixth row?

Remember multi-dimensional dynamic arrays (2016-03-03:195–197)

1. `double *matrix = (double *)malloc(sizeof(double)*10*4);`
   `assert( matrix != NULL );`
   allocates $10 \cdot 4 \cdot$ sizeof(double) bytes (likely to be 320 bytes; but not certain)
2. `matrix[5*4+1]`
   (`matrix[1*10+5]` can also be justified)

Extracting Information from a Matrix in C

Problem Consider the following code and add the missing code that allocates the space for the matrix.

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>

#define SIZE 17 /* number of rows and number of columns both equal to SIZE */

int main(void) {
  int i, j;
  double **m;

  /* allocate space for SIZE x SIZE matrix m here */

  for ( i=0; i<SIZE; i++ ) {
    for ( j=0; j<SIZE; j++ ) {
      m[i][j] = rand();
    }
  }
  return EXIT_SUCCESS;
}
```

```
m = (double **)malloc(sizeof(double *)*SIZE);
assert( m!=NULL );
for ( i=0; i<SIZE; i++ ) {
  m[i] = (double *)malloc(sizeof(double)*SIZE);
  assert( m[i]!=NULL );
}
```

Reductions

Problem In the knapsack problem you are given a knapsack with weight limit $W \in \mathbb{N}$ and a sequence of items, each with its own integer weight and value. You want to fill your knapsack with a selection of items so that the contents of your knapsack has as much value as possible but does not exceed the weight limit W. In the optimisation version of this problem the best possible value is sought. In the decision version of this problem you are additionally given a number k and have to decide if a value of at least k can be achieved. Describe how the optimisation version can be solved using the decision version (i. e., describe a reduction from the optimisation version to the decision version).

Remember different kinds of optimisation problems and reductions (2016-04-11:272–275)

Idea Items with value ≤ 0 or weight $> W$ play no part in optimal solution.
We can discard them. Let the remaining items have values $v_1, v_2 \ldots, v_n$ and weights $w_1, w_2 \ldots, w_n$.
We know optimal solution has value at least $\max_n\{v_1, \ldots, v_n\}$.
We know optimal solution has value at most $\sum_{i=1} v_i$.
Perform binary search between these two boundaries.

Reductions

Problem Knapsack problem: weight limit W, list of items with weights and values (all weights and values are integers)
Problem Find items that maximise value and respect weight limit W.
Opt. Problem Find optimal value.
Dec. Problem Decide if solution with value k exists.
To do Reduce optimisation problem to decision problem.

Remember different kinds of optimisation problems and reductions (2016-04-11:272–275)

Solution

Discard all items with value ≤ 0 or weight $> W$.
Let remaining items have values $v_1, v_2, \ldots, v_n$ and weights $w_1, w_2, \ldots, w_n$.
Let lower $:= \max\{v_1, \ldots, v_n\}$. Let upper $:= \sum_{i=1}^{n} v_i$.
While lower $<$ upper do
  middle $:= \lceil (\text{lower} + \text{upper})/2 \rceil$
  If solution with value middle exists then lower $:=$ middle else upper $:=$ middle $- 1$.
Output 'optimal solution has value middle'.

Halting Problem

Problem Imagine you are talking to a computer science student who does NOT know what computability is and what the halting problem is. Explain exactly in plain English what the non-computability of the halting problem means. What exactly cannot be done?

Remember computability and halting problem (2016-04-14:295–299)

Solution Nobody is able and nobody will ever be able to write a program that takes as input the source code of a program and an input for it and that will output if that program will get stuck forever on that input or not.

Other Non-Computable Problems

Problem Consider the following computational problem.
Input is a Turing machine (e. g., a program for the simulator from the practicals).
Output is YES if there exists an input for the Turing machine that makes the Turing machine run for ever and
NO if the Turing machine stops for all inputs.
Describe precisely how you can prove this problem to be non-computable by making use of a reduction and the fact that the halting problem is not computable.

Remember reductions (2016-04-11:275)

Plan Prove 'halting problem $\leq$ this problem'
(since this proves that this problem is not computable because otherwise we could solve the halting problem with the help of this problem and the '$\leq$-algorithm'

Other Non-Computable Problems

Problem Input is a Turing machine.
Output: Decide if there exists an input that makes the Turing machine run for ever.

Plan Prove 'halting problem $\leq$ this problem'

Remember input for halting problem is TM P and input I

Define TM M that does the following:

1. Erase any input that M has initially.
2. Write I on the tape.
3. Simulate P on I.

Use M as input for 'this problem' and use received answer as answer

Observations

- M behaves for any input like P on I (because ignores input).
- M stops on all inputs if and only if P stops on I.
- Answer for M is correct answer for P and I.
- '$\leq$-algorithm' and 'this problem' together solve halting problem
- '$\leq$-algorithm is easy to compute, halting problem is not computable, therefore 'this problem' is not computable

About Page Rank

Problem Point out all mistakes in the following text about Page Rank.

Page Rank is an algorithm due to Sergey Brin that ranks web pages by how important Google thinks they are. It's a specialised algorithm for this problem only. It's modelled on careful traffic analysis and has no parameters. The exact idea is too difficult to explain and this is even more true for the details. The only and easy way to manipulate its results is to have loads of links from each page to the target page. Page Rank still defines Google's ranking the same way it did 1998.

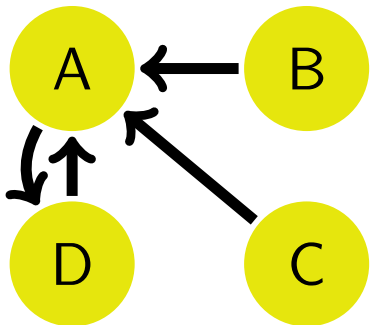
Remember Page Rank (2016-04-18/21:305–322)

List of mistakes and corrections

- Page Rank has its name because it is by Larry Page.
- Importance is based on the number of unique incoming links and the page rank of those sites.
- Page Rank has applications in many different areas.
- It's modelled on a random surfer's behaviour.
- The restart probability r is a parameter.
- The main idea is simple (see 2^{nd} point).
- $\mathrm{PR}(v)_0 := 1/n; \mathrm{PR}(v)_t := (r/n) + (1-r) \sum_{w \in I(v)} \mathrm{PR}(w)_{t-1}/L(w)$
- Page Rank only counts unique links (ignores multiple links).
- Many pages linking to the same page can manipulate Page Rank results.
- Google updated the page ranking multiple times.

Page Rank

Problem For the graph below and $r = .15$, compute the values that Page Rank obtains in the first round after initialisation.



Remember Page Rank algorithm (2016-04-21:318–322)

Initialisation $\mathsf{PR}(A) = 1/4, \mathsf{PR}(B) = 1/4, \mathsf{PR}(C) = 1/4, \mathsf{PR}(D) = 1/4$

1st round

$$\mathsf{PR}(A) = \frac{0.15}{4} + 0.85 \left(\frac{\mathsf{PR}(B)}{1} + \frac{\mathsf{PR}(C)}{1} + \frac{\mathsf{PR}(D)}{1} \right)$$
$$= \frac{0.15}{4} + 0.85 \cdot 3 \cdot \frac{1}{4}$$
$$\mathsf{PR}(B) = \frac{0.15}{4}$$
$$\mathsf{PR}(C) = \frac{0.15}{4}$$
$$\mathsf{PR}(D) = \frac{0.15}{4} + 0.85 \cdot \frac{\mathsf{PR}(A)}{1} = \frac{0.15}{4} + \frac{0.85}{4}$$

Summary & Take Home Message 'Exam Preparation'

Things to remember

- exam contributes 30% $\Rightarrow$ no panic
- numbers
- matrices and arrays
- computability and reductions
- Page Rank

Take Home Message

- Don't panic.
- Please, prepare for the exam.
- Not all topics are relevant for the exam. What is relevant is covered in the two revision lectures (25th and 28th of April 2016).